UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

DESTINATION PREDICTION OF CAR SHARING AND OTHER VEHICLES USING DEEP LEARNING

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Xavi Roig Aznar

In partial fulfilment

of the requirements for the degree in

Science and Technology of Telecommunications
ENGINEERING

Advisors: Elisa Sayrol Clols and José Adrián Rodríguez Fonollosa

Barcelona, June 2019

Abstract

Noise and environmental pollution are one of the main problems in the city of Barcelona. That is why the research group CARNET is planning this project to try to reduce the emissions of gases produced by taxi and car sharing sectors, as well as to improve the distribution of taxis in the city to avoid more kilometers than necessary and reduce customer wait times.

To achieve these goals, a recurrent neural network model has been developed that through the vehicle's origin point and some additional metadata, the system is able to predict the vehicle's destination point.

After training the model, the values of the parameters of the optimal system are indicated.

Resum

La contaminació acústica i medi ambiental és un dels principals problemes que hi ha a la ciutat de Barcelona. És per això que el grup de recerca CARNET planteja aquest projecte per tal d'intentar reduir les emissions de gasos produïdes pels sectors del taxi i del *car sharing*, així com per millorar la distribució dels taxis a la ciutat per tal d'evitar realitzar més kilòmetres dels necessaris i reduir els temps d'espera dels clients.

Per aconseguir aquests objectius, s'ha desenvolupat un model de xarxa neuronal recurrent que mitjançant el punt d'origen el vehicle i algunes metadades addicionals, el sistema general és capaç de predir el punt de destí del vehicle.

Després d'entrenar el model, els valors dels paràmetres del sistema òptims són indicats.

Resumen

La contaminación acústica y medioambiental es uno de los principales problemas que hay en la ciudad de Barcelona. Es por ello que el grupo de investigación CARNET plantea este proyecto para intentar reducir las emisiones de gases producidas por los sectores del taxi y del *car sharing*, así como para mejorar la distribución de los taxis en la ciudad para evitar realizar más kilómetros de los necesarios y reducir los tiempos de espera de los clientes.

Para conseguir estos objetivos, se ha desarrollado un modelo de red neuronal recurrente que mediante el punto de origen del vehículo y algunos metadatos adicionales, el sistema general es capaz de predecir el punto de destino del vehículo.

Después de entrenar el modelo, los valores de los parámetros del sistema óptimo son indicados.

Agraïments

Donar les gràcies als tutors d'aquest treball Elisa Sayrol Clols i José Adrián Rodríguez Fonollosa per la seva dedicació i confiança durant aquests mesos.

També agrair a la família i amics el seu suport durant el desenvolupament d'aquest projecte i durant el transcurs del grau en general.

Gràcies per tot!

Historial de revisions i registre d'aprovació

Revisió	Data	Propòsit
0	10/06/2019	Creació del document
1	25/06/2019	Revisió del document

LLISTA DE DISTRIBUCIÓ DEL DOCUMENT

Nom	e-mail
Xavi Roig Aznar	xavi.roig97@gmail.com
Elisa Sayrol Clols	elisa.sayrol@upc.edu
José Adrián Rodríguez Fonollosa	jose.fonollosa@upc.edu

Escrit per:		Revisat i aprovat per:	
Data	10/06/2019	Data	25/06/2019
Nom	Xavi Roig Aznar	Nom	Elisa Sayol Clols i José Adrián Rodríguez Fonollosa
Posició	Project Autor	Posició	Supervisor

Índex

Índex de Figures

Índex de Taules

1. Introducció

1.1. Objectius del projecte

Cada cop més gent s'està desplaçant des dels entorns rurals a les grans metròpolis degut a l'oferta de treball i al conjunt d'activitats d'oci que es poden realitzar actualment a les grans ciutats, com és el cas de Barcelona. Però, tot i això, tot no són avantatges ja que aquest gran creixement de població resident a les grans ciutats provoca un conjunt de conseqüències negatives.

Algunes de les principals conseqüències provocades per aquest excés de població són l'augment de la dificultat a l'hora de desplaçar-se amb qualsevol tipus de vehicle pels carrers de la ciutat així com la contaminació, tant acústica com medi ambiental, provocada per aquests.

Aquests factors han provocat que el grup de recerca CARNET es plantegés realitzar aquest projecte per tal d'estudiar, investigar i trobar solucions als diferents problemes que es troben les companyies de taxi i *car sharing* diàriament a la ciutat de Barcelona.

El grup de recerca CARNET està compost pels grups de recerca de Seat i Volkswagen així com per la Universitat Politècnica de Catalunya (UPC). CARNET és un centre obert que treballa en la recerca i la innovació en les àrees de l'automobilisme i la mobilitat. El seu treball es basa en realitzar projectes que aportin innovacions i solucions que permetin tancar l'escletxa que hi ha entre la recerca acadèmica i la innovació industrial en la mobilitat urbana.

El projecte plantejat pel grup CARNET que es descriu en aquesta memòria, té com a objectiu predir el destí del taxi o vehicle de *car sharing* a partir del seu punt d'origen i de diferents metadades.

Per arribar a l'objectiu plantejat, aquest projecte utilitza diferents algorismes i tècniques basades en Deep Learning (DL). Aquests algorismes ajuden a intentar trobar solucions als diferents problemes que es troben els taxis i els *car sharing* a la ciutat de Barcelona per tal de:

- Trobar una distribució adequada dels vehicles a Barcelona
- Reduir els kilòmetres innecessaris a realitzar
- Reduir la congestió del tràfic
- Reduir les emissions de gasos
- Reduir el temps d'espera dels clients

Com en qualsevol projecte de DL, es necessària una base de dades per tal de poder entrenar el sistema. Aquesta base de dades ha estat proporcionada pel mateix grup CARNET. Consta de les diferents carreres de taxi a la ciutat de Barcelona durant el transcurs d'un any natural. Aquesta base de dades està composta per un total d'aproximadament 250000 carreres.

La forma de desenvolupament del projecte per a arribar a la solució final del problema és molt variada i en aquest projecte, s'ha escollit utilitzar xarxes neuronals recurrents.

La utilització de xarxes neuronals recurrents implica la necessitat de tenir seqüències d'entrada, és a dir, tenir com a entrada de la xarxa un conjunt de parells de punts de coordenades origen-destí ja que aquest conjunt de punts de coordenades permetran analitzar el comportant d'un taxi durant un conjunt carreres consecutives i, d'aquesta forma, la xarxa aprendrà a predir el punt de destí del taxi a partir, no només de les condicions inicials de la carrera actual, sinó també, de les dades que caracteritzen les carreres anteriors a l'actual.

1.2. Requeriments i especificacions

Aquest projecte s'ha desenvolupat íntegrament amb el llenguatge de programació Python i amb el *framework* Pytorch per tal de poder implementar el model de xarxa neuronal recurrent. També s'han utilitzat diferents llibreries com *sklearn* i *numpy*.

Finalment, també s'ha fet ús d'una Unitat Gràfica de Processament (GPU) per tal de realitzar l'entrenament i l'avaluació del model de xarxa d'una forma més ràpida.

Tot aquest projecte s'ha desenvolupat dins dels servidors del Grup d'Imatge de la UPC ja que facilitaven tots els requeriments enunciats en els dos paràgrafs anteriors.

1.3. Pla de treball

1.3.1. Paquets de treball

WP 1: Estat de l'art (recerca)

WP 2: Embedding layer

WP 3: Attention Module

WP 4: RNN Module

WP 5: Mòdul de predicció

WP 6: Ajustament d'hiperparàmetres i millora del sistema

WP 7: Documentació

1.3.2. Diagrama de Gantt

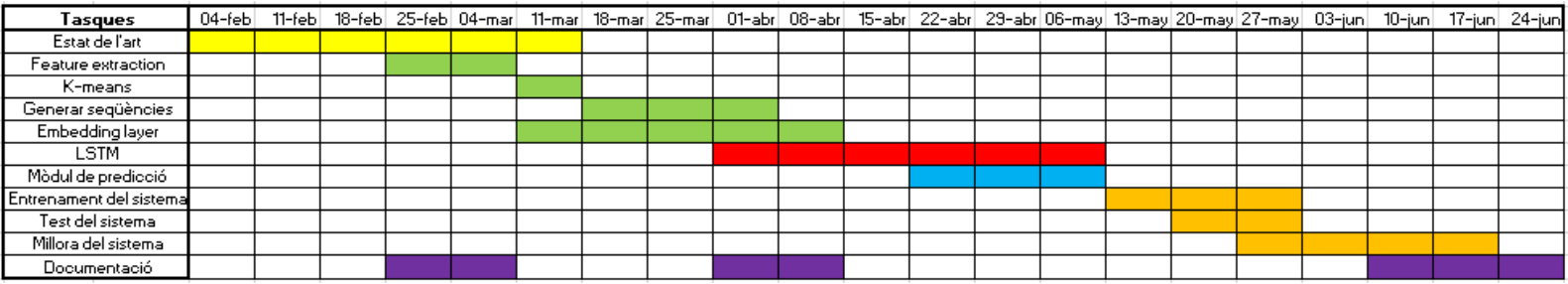


Figura 1.1.1: Diagrama de Gantt

1.3.3. Incidències i modificacions

Al pla de treball inicial es pretenia implementar un mòdul d'atenció que permetés millorar els resultats obtinguts però per falta de temps no ha estat possible la implementació d'aquest mòdul al sistema general.

2. Estat de l'art d'anteriors estudis i algorismes utilitzats

En aquesta secció del document s'explica la diferent informació apresa per poder realitzar aquest projecte. Aquesta informació inclou tant la part de lectura d'articles d'altres projectes que treballen sobre estudis similars a aquest com la part teòrica dels algorismes i tecnologies utilitzades durant el desenvolupament del projecte.

2.1. Estudis anteriors relacionats amb prediccions de localitzacions

En estudis anteriors relacionats amb la predicció de localitzacions o de mobilitat s'han utilitzats diferents tipus d'algorismes i tècniques [1] [2] [3] [4], des de Models Ocults de *Markov* fins a xarxes neuronals.

En una competició de *kaggle* de l'any 2015 [5], el model guanyador utilitzava una xarxa MLP (*Multi Layer Perceptron*) entrenada sobre diferents punts de la trajectòria del taxi i utilitzant metadades com l'identificador del conductor, el temps de sortida...

Aquest sistema, però, té diferents inconvenients [6] com per exemple que es necessiten totes les dades del trajecte i que, per tant, per predir el punt de destí és necessària gairebé saber tota la trajectòria del vehicle.

En conseqüència, s'han dut a terme avenços basats en les xarxes neuronals recurrents [7], que permeten predir el punt de destí del trajecte del taxi només tenint en compte el punt de sortida i diferents metadades (dades del taxi i dades temporals) [8] i fixant-se en la seqüència dels últims llocs visitats.

2.2. Algorisme K-means

Els algorismes no supervisats són aquells algorismes de Machine Learning on durant l'entrenament només hi ha vectors d'entrada, ja que aquests vectors no tenen cap referència (*target*). És a dir, mentre es duu a terme l'entrenament, els vectors d'entrada no tenen cap objectiu perfectament definit. Aquest tipus d'algorismes tenen un caràcter exploratori ja que l'únic que es pot fer és descriure l'estructura de dades per tal d'intentar trobar algun tipus d'organització i simplificar l'anàlisi.

Aquest tipus d'algorismes s'utilitzen principalment per a problemes de *clustering*, agrupaments de co-ocurrència i *profiling*.

L'algorisme K-means pertany al primer tipus (*clustering*) i permet agrupar el conjunt de dades en K classes diferents. Aquest algorisme és un algorisme iteratiu el qual consisteix en una inicialització aleatòria dels centroides de les K classes i una actualització del valor d'aquests centroides fins arribar al punt de convergència.

A continuació s'adjunta un petit exemple, per a K=2, del conjunt de fases d'aquest algorisme per explicar de forma més detallada i visual cadascun dels diferents passos.

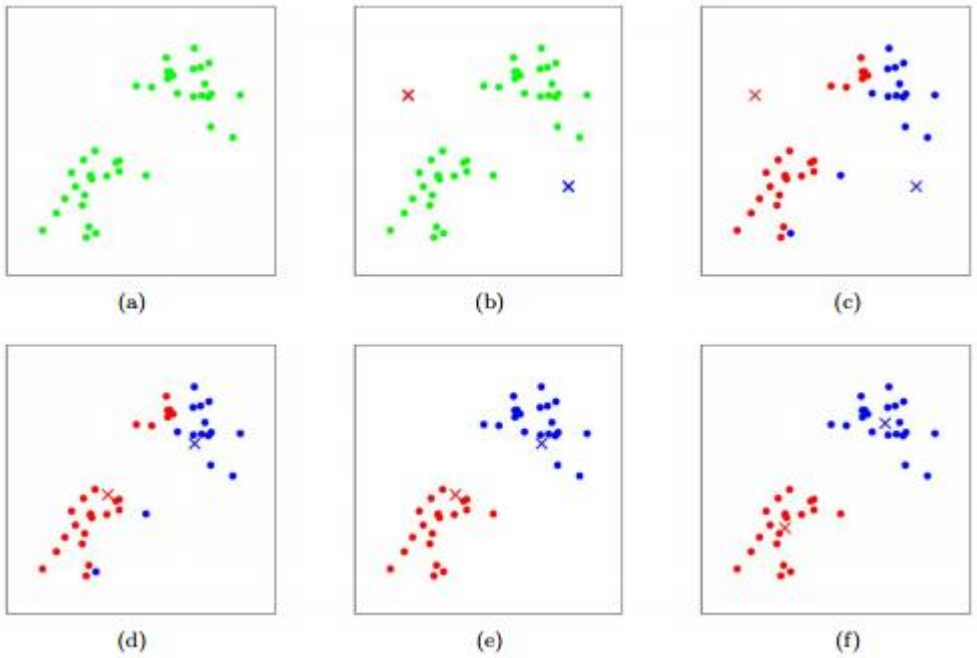


Figura 2.1: Fases de l'algorisme K-means

(a) Es representa un conjunt de dades
(b) S'inicialitzen els 2 centroides de forma aleatòria
(c) Es generen 2 grups, assignant cada punt a la classe amb distància menor
(d) Actualització del valor dels centroides
(e) Repetició dels passos (c) i (d)
(f) Repetició dels passos (c) i (d)
(g) Convergència de l'algorisme

Finalment, es comprova que els valors dels centroides són iguals als de (f) ja que s'ha arribat a la convergència de l'algorisme.

2.3. Xarxes Neuronals Recurrents

Les persones no pensen des de zero cada segon. Cada paraula que estàs llegint l'estàs entenent basant-te en les paraules que la precedeixen. Aquesta acció natural dels humans, les xarxes neuronals tradicionals no la poden dur a terme. És per això que les xarxes neuronals recurrents van ser creades, per resoldre els diferents problemes on és necessari tenir en compte el context temporal i, per tant, tenir memòria. [9]

Aquest tipus de xarxes són xarxes que tenen bucles dins d'elles, fet que permet que tinguin memòria. Tot i semblar molt més complexes, són bastant similars a les xarxes neuronals tradicionals, només cal pensar que aquest tipus de xarxes són com múltiples còpies de xarxes tradicionals on cada xarxa li passa la seva informació a la xarxa posterior, és a dir, tenen molta relació amb informacions seqüencials [9].

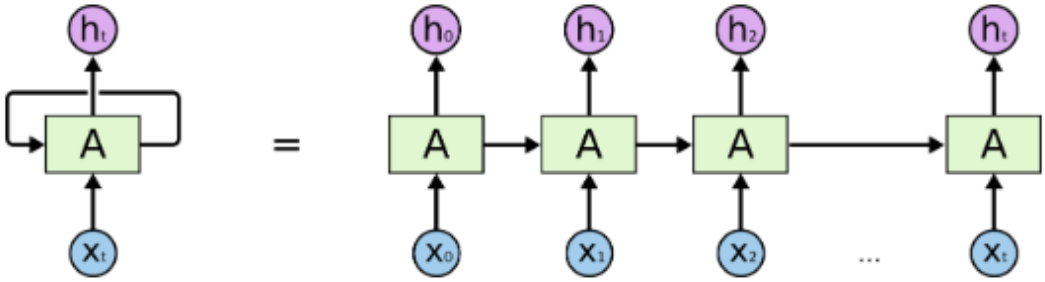


Figura 2.2: Xarxa Neuronal Recurrent

2.3.1. Arquitectura

En aquest projecte s'utilitza una xarxa neuronal recurrent composta per:

- Embedding layer
- Capa recurrent
- Capa lineal
- Softmax

2.3.1.1. Embedding layer

La funció de la capa d'Embedding no és altra que passar d'un espai que té una dimensió per característica a un espai vectorial continu de menys dimensions.

Una *Embedding layer* és una matriu de pesos, que precedeix a la xarxa neuronal, on els pesos es van actualitzant durant l'entrenament d'un model de xarxa neuronal. Aquesta matriu converteix un conjunt de característiques en vectors de mida fixa, anomenats *latent vectors*. És a dir, aquesta matriu està formada per un conjunt de *latent vectors*, on cadascun d'aquests representa una característica.

Cadascuna de les files de la matriu està associada al seu índex i representa una característica, és a dir, si es té un vocabulari de 2000 característiques diferents, la llista d'índexs serà: [0,1,....,1999].

L'origen dels *embeddings* es correspon als *words embeddings* [10], com és, per exemple, el word2vec, però en aquest cas aquest algorisme no ens interessa.

La característica descrita pels vectors, pot tenir qualsevol format (no té perquè ser una paraula) i el que permeten aquests *embeddings* és compactar la informació i trobar relacions entre diferents característiques que potser no sembla que tinguin una relació clara.

D'altra banda, existeix el *one-hot encoding* que té la mateixa funció que l'*embedding layer* però treballa de forma diferent. El *one-hot encoding* consta d'un vector de dimensió igual al número de dades amb tots els seus elements iguals a zero excepte l'element que descriu la dada corresponent, que té valor 1.

Llavors aquest vector es multiplica per la matriu de *latent vectors* i el resultat és l'*embed* de la dada corresponent.

Es podria dir que el *one-hot encoding* és un algorisme on el vector de zeros i un u s'ha de multiplicar per la matriu mentre que l'*embedding layer* és un algorisme de selecció on es selecciona la fila de la matriu corresponent a l'índex de la dada corresponent. Per tant, hi ha una gran diferència computacional entre aquests dos algorisme que fa que quan es treballa amb *datasets* molt grans, s'utilitzi l'*embedding layer*.

Un exemple que explica clarament el funcionament d'aquesta capa d'*embedding* és el següent:

"deep learning is very deep"

Es crea un vocabulari:

Vocabulari	**Índex**
deep	1
learning	2
is	3
very	4

Taula 2.1: Relació dada-índex

L'*input* passa a ser: [1,2,3,4,1]

Llavors, l'*Embedding matrix* (amb vectors de longitud 6) queda definida de la següent forma:

ÍNDEX	***LATENT VECTORS***					
1	0.32	0.02	0.48	0.21	0.56	0.15
2	0.65	0.23	0.41	0.57	0.03	0.92
3	0.45	0.87	0.89	0.45	0.12	0.01
4	0.65	0.21	0.25	0.45	0.78	0.82

Taula 2.2: Embedding matrix

La paraula "*deep"* queda representada pel vector: [0.32,0.02,0.48,0.21,0.56,0.15]

L'*embedding size* és la longitud dels vectors que defineixen cadascuna de les característiques i és definida amb la longitud que es cregui convenient.

És evident que en el cas de dues característiques similars, els seus *latent vectors* tindran valors molt semblants i, per tant, aquestes dues característiques estaran molt pròximes a l'espai X-dimensional definit pels vectors (X igual a la longitud dels vectors). La següent imatge és un bon exemple:

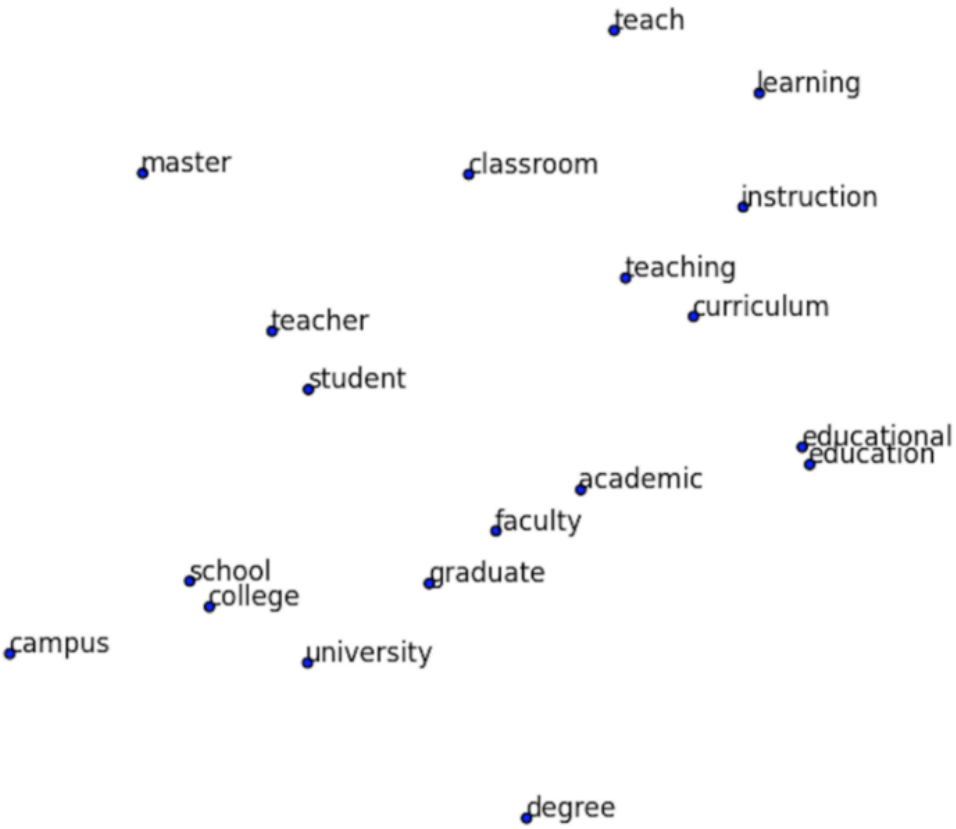


Figura 2.3: Representació espacial de les característiques

2.3.1.2. Long Short-Term Memories (LSTM)

Tot i que les RNNs són xarxes neuronals que disposen de memòria, aquesta memòria és una memòria "curta" ja que si la predicció que s'ha de realitzar depèn d'un context molt anterior aquestes xarxes és molt difícil que siguin capaces d'aprendre a relacionar aquesta informació.

Tot i això, les LSTMs no tenen aquest problema ja que estan dissenyades per evitar aquest problema de llarga dependència.

La peculiaritat de les LSTMs està en la seva unitat bàsica que està formada per 4 capes neuronals diferents:

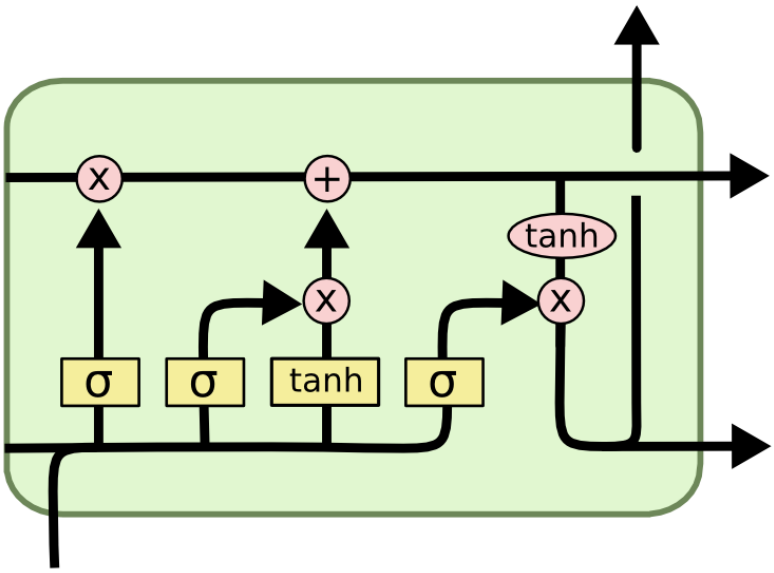


Figura 2.4: Unitat bàsica de la LSTM

La clau de les LSTMs és l'estat de la cèl·lula (línia horitzontal superior). La LSTM té l'habilitat d'afegir o eliminar informació.

Aquest procés està sempre regulat per unes estructures anomenades portes. Cada unitat bàsica està composta per: la porta d'oblit, la porta d'entrada i la porta de sortida.

Les equacions realitzades dins de cada unitat bàsica són les següents [9]:

$$\begin{aligned} f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\ h_t &= o_t \circ \sigma_h(c_t) \end{aligned} \tag{2.1}$$

2.3.1.3. Capa lineal i Softmax

En un problema de classificació, la combinació d'aquestes dues capes permet obtenir la probabilitat de pertinença, a cadascuna de les classes de sortida, de l'element d'entrada de la xarxa.

La probabilitat de pertànyer a cadascuna de les K classes de sortida s'obté a partir de la següent funció:

$$P_i = \frac{exp(e_i)}{\sum_{j=i}^{C} exp(e_j)} \tag{2.2}$$

En aquest projecte, la sortida del sistema general han de ser les coordenades de destí, latitud i longitud, tot i que les dades espacials d'entrada del sistema no són les coordenades exactes d'origen de la carrera sinó que és un dels K clústers en els que es dividida la ciutat de Barcelona a partir de l'algorisme K-means. Això significa que després d'aquestes dues capes mencionades en aquest apartat, ha d'haver-hi un mòdul que a partir de les probabilitats del Softmax sigui capaç de predir les coordenades del lloc de destí.

3. Desenvolupament del projecte

En aquesta secció del document s'expliquen tots els passos que s'han dut a terme per tal d'obtenir els resultats que s'exposen a la secció següent. És a dir, s'explicaran detalladament totes les parts de l'arquitectura del model implementat per a l'estudi de la predicció de les coordenades del punt de destí de qualsevol taxi o *car sharing* a la ciutat de Barcelona.

Aquesta arquitectura consta de tres submòduls principals: (i) Extracció de les característiques i Embedding, (ii) Model Recurrent (LSTM) i (iii) Predicció de les coordenades [8].

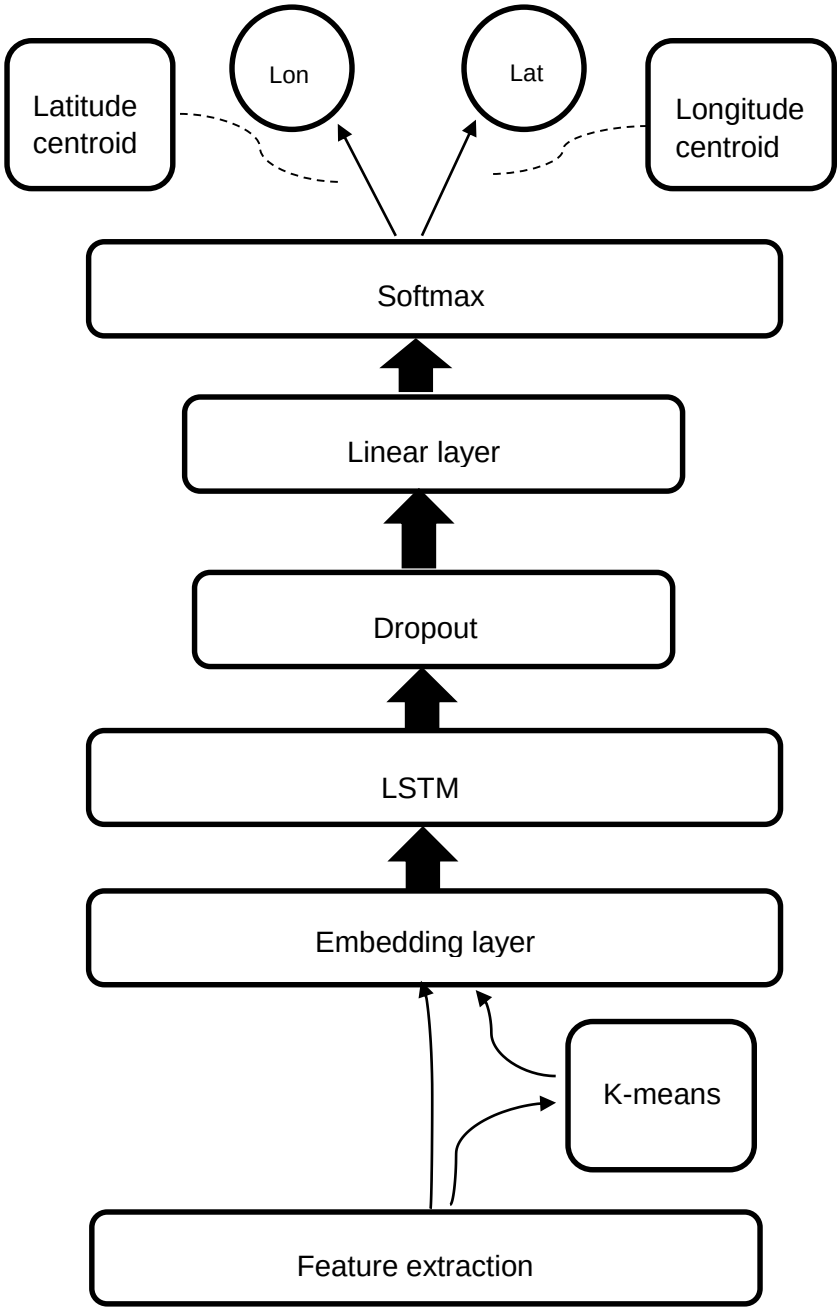


Figura 3.1: Model general

3.1. Extracció de les característiques de la base de dades

Tal i com s'ha explicat a la secció d'Introducció, la base de dades ha estat facilitada pel grup CARNET. Cadascuna de les entrades del *dataset* conté diferents tipus d'informació i no tota aquesta informació és útil per al desenvolupament d'aquest projecte.

Per tant, el primer que s'ha fet ha estat un filtratge de les característiques de la base de dades per a quedar-nos només amb les característiques espacials i temporals i amb els identificadors del taxi i del taxista.

D'aquesta manera, tota la informació necessària per a la realització del projecte ha quedat guardada de la següent forma:

	Identificadors	**Característiques Temporals**	**Característiques Espacials**
Carrera 1	ID_1	T_1	$Coordenades_1$
...	...	...	...
Carrera N	ID_N	T_N	$Coordenades_N$

Taula 3.1: Base de dades

En total, s'han extret un total de catorze característiques, a continuació enumerades:

- Identificador del taxi
- Identificador del taxista
- Jornada
- Dia de la setmana
- Data d'inici
- Hora d'inici
- Minut d'inici
- Data final
- Hora final
- Minut final
- Latitud inicial
- Longitud inicial
- Latitud final
- Longitud final

Totes aquestes característiques, com s'ha dit anteriorment, són útils per al projecte, però no totes tenen la mateixa funció ni s'utilitzen al mateix mòdul del model general.

Això s'explica fàcilment en el cas de la *jornada*, que serveix per fer un preprocessat de la informació per tal de generar les seqüències d'entrada de la xarxa recurrent tal i com s'explica a l'apartat 3.4.

Per altra banda, les últimes dues característiques (latitud final i longitud final) són les anomenades *targets* ja que seran les coordenades amb les quals es comparará la

predicció del sistema per tal de computar la *loss* del model i així realitzar el *Backpropagation* per dur a terme l'actualització dels paràmetres del model.

Finalment, cal comentar que, com en qualsevol projecte de Deep Learning, el conjunt de dades es divideix en tres conjunts diferents, cadascun dels quals té una funció determinada en el desenvolupament del projecte:

- Conjunt d'entrenament: Aquest conjunt serveix per entrenar la xarxa recurrent i trobar el model amb millors resultats. Se li ha assignat un 75% de les dades.
- Conjunt de validació: Aquest conjunt serveix per veure que la xarxa no genera *overfit* o *underfit* respecte el conjunt de dades d'entrenament. Se li ha assignat un 15% de les dades.
- Conjunt de test: Aquest conjunt serveix per testejar el model final i obtenir els resultats amb unes dades que mai han estat vistes per ella. Se li ha assignat un 10% de les dades.

3.2. Algorisme K-means

El funcionament de l'algorisme K-means s'ha explicat a l'Estat de l'Art, per tant, ara s'explicarà per a què s'ha utilitzat aquest algorisme en aquest projecte, amb quin conjunt de dades s'ha utilitzat i quins resultats s'han obtingut.

En aquest cas, aquest algorisme s'ha utilitzat per a crear un conjunt de clústers que permetin distribuir de manera adequada el conjunts de punts de coordenades de la base de dades a la ciutat de Barcelona.

Tal i com s'ha explicat a l'Estat de l'Art, aquest algorisme arriba a la convergència, i és per això que a la gràfica de valors amb diferents número de clústers existeix sempre un "punt de colze".

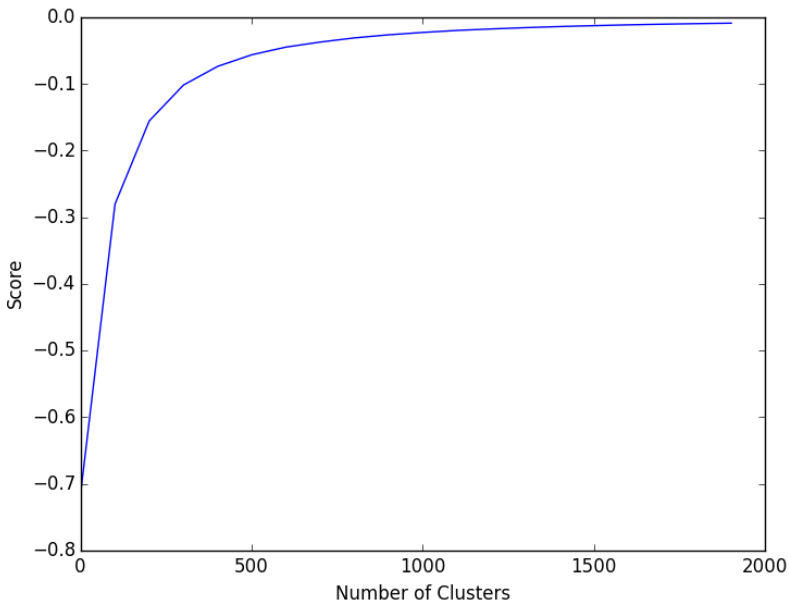


Figura 3.2: Convergència algorisme K-means

A partir de la gràfica anterior, s'ha escollit un valor de K = 300 ja que s'observa que a partir d'aquest número de clústers, el valor de l'algorisme K-means disminueix molt lentament fins a arribar a la convergència.

La imatge que es mostra a continuació mostra un conjunt de valors coordenades i de centroides creats per l'algorisme:

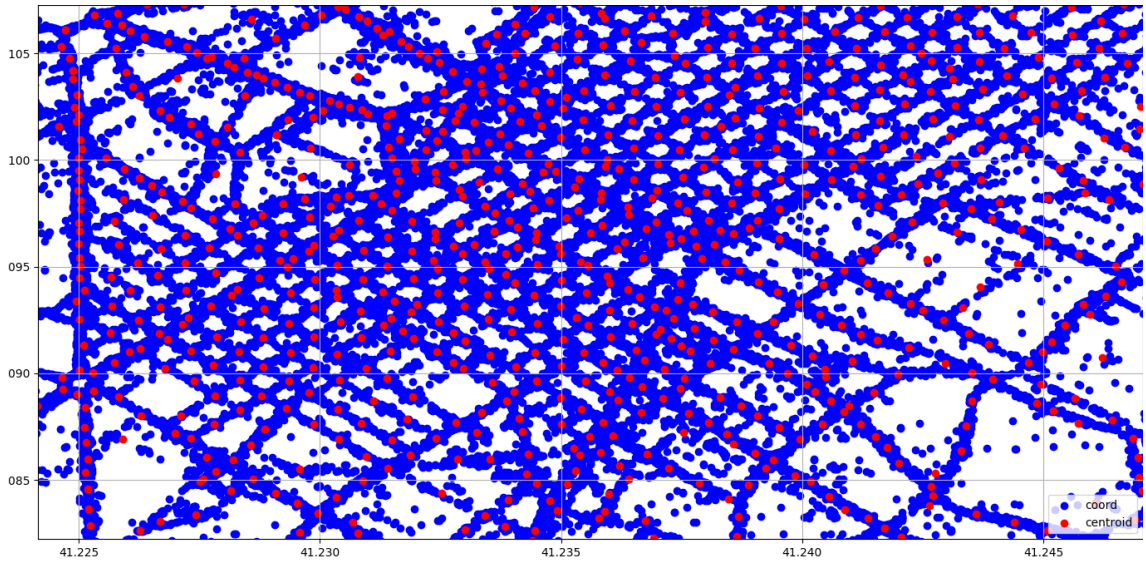


Figura 3.3: Distribució clústers respecte a les coordenades

Aquest clústers descriuen el contingut espacial de la carrera del taxi, és a dir, cada parell de coordenades d'origen de cada carrera de taxi, ha estat assignat al clúster amb el centroide més pròxim. Llavors, el vector corresponent a la fila de l'*Embedding matrix* amb índex igual al clúster ha passat a formar part de l'entrada de la xarxa.

A continuació es mostra de forma gràfica com es realitza aquest pas durant l'execució del procés d'entrenament del model:

Latitud	Longitud
41.235E	2.089N

Clúster	Centroid_lat	Centroid_lon
0	41.234E	2.09N
1	41.35E	2.15N
2	41.32E	2.05N
...	...	...
1999	41.236E	2.075N

Taula 3.2:Relació índex-coordenades

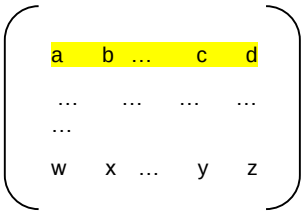


Figura 3.4: Embedding matrix

Com es pot observar, aquest parell de coordenades pertanyen al clúster 0, per tant, el *latent vector* seleccionat de la matriu d'*embedding* serà el primer.

Al següent subapartat s'explica més detalladament tot el referent a la part de la capa d'*embedding* relacionant tot l'explicat a l'apartat de l'Estat de l'Art amb el model implementat en aquest projecte.

3.3. Embedding layer

Al primer subapartat d'aquesta secció ja s'ha comentat breument quines són les característiques d'entrada de la xarxa recurrent (*taxiID, driverID, dia, data, hora, minut, clúster)* i, per tant, aquestes són les característiques a les quals se'ls ho ha de dur a terme *l'embedding* per tal de fer més compacta la seva entrada a la xarxa.

El primer que s'ha fet ha sigut crear una relació característica-índex per a cadascuna de les característiques d'entrada de la xarxa recurrent de tal forma que quedi una taula similar a la següent (per al cas del *taxiID*):

taxiID	**Índex**
3	2
8	3
17	4
...	...
39	*N-2*
48	*N-1*
56	*N*

Taula 3.3: Identificador del taxi

Un detall molt important no comentat anteriorment és el fet que el primer índex de la taula anterior és el 2 i no el 0, això és així degut a que els índexs 0 i 1 són reservats per als valors de *padding* i de valors de característiques desconeguts, respectivament.

característica	**índex**
padded	0
unknown	1

Taula 3.4: Valors padded i unknown

És a dir, una característica tindrà un valor *padded* quan s'hagi de completar un conjunt de carreres de longitud fixa amb característiques amb índex 0. Aquest fet és l'anomenat *padding*.

En canvi, quan es realitza la inferència, poden haver característiques amb nous valors que no apareguin al conjunt d'entrenament. A qualsevol d'aquests valors se li assigna l'índex 1, d'aquesta manera la xarxa ja sap que aquella característica d'entrada és nova i no estava al conjunt de dades d'entrenament ni de validació. Aquest índex, per tant, no serà mai assignat a característiques que tinguin un rang de valors finit, com per exemple la hora ja que *hora* ϵ *[0,23]*.

Com ja s'ha explicat a la secció d'*Estat de l'art*, aquests índexs es converteixen en vectors (*latent factors*) de mida fixa que es van actualitzant en cada iteració de l'entrenament del model complet.

Llavors, a cadascuna de les característiques del *dataset*, li correspon una *embedding* matrix independent de la resta de característiques per representar-la i finalment es concatenen tots els *embedded vectors* dels índexs corresponents de cada característica. Aquest conjunt de vectors concatenats són l'entrada de la xarxa recurrent. A la següent figura d'exemple queda il·lustrat amb més claredat com es realitza aquest pas.

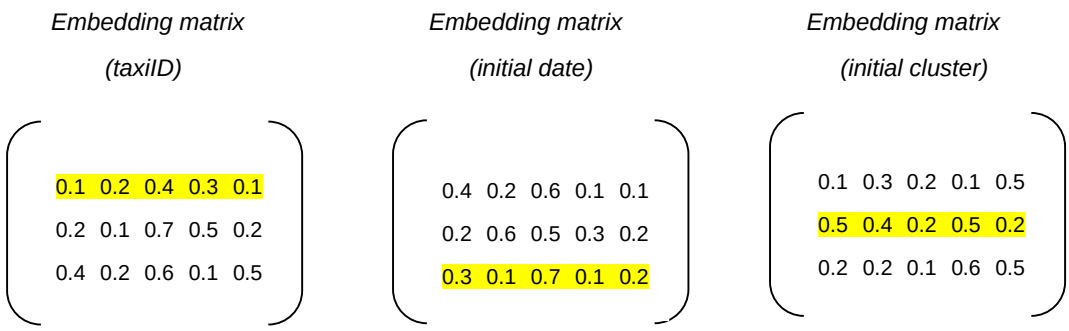


Figura 3.5: Input de la xarxa neuronal

Suposant que només tinguéssim aquestes tres característiques i que cadascuna només tingués tres elements i l'*embedding size* fos igual a cinc, i els índexs d'entrada fossin [1,3,2], respectivament, l'entrada de la xarxa recurrent seria:

[0.1,0.2,0.4,0.3,0.1,0.3,0.1,0.7,0.1,0.2,0.5,0.4,0.2,0.5,0.2]

Les dimensions de les *embedding matrix* de cadascuna de les 7 característiques d'entrada de la xarxa recurrent són les següents:

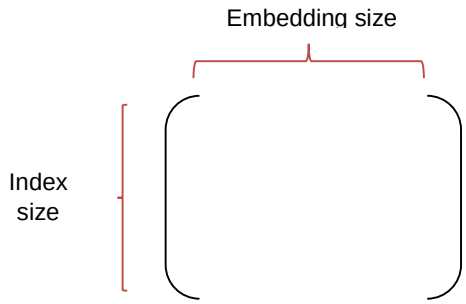


Figura 3.6: Dimensions de la Embedding Matrix

Index size: mida del conjunt d'índexs de la característica.

Embedding size: longitud dels *latent vectors* que descriuran a cadascun dels índexs de la característica.

Evidentment, aquesta matriu és tridimensional ja que té una profunditat de la mida del *batch*.

Per als identificadors i per a les característiques espacials, l'*embedding size* és de mida 10 mentre que per als clústers és de mida 20.

3.4. Generació de les seqüències de característiques

Cadascuna de les mostres del *batch* ha d'estar formada per un conjunt de parelles origen-destí amb les seves respectives característiques. És a dir, cadascun dels *batches* de la xarxa té la forma que mostra la següent imatge [8]:

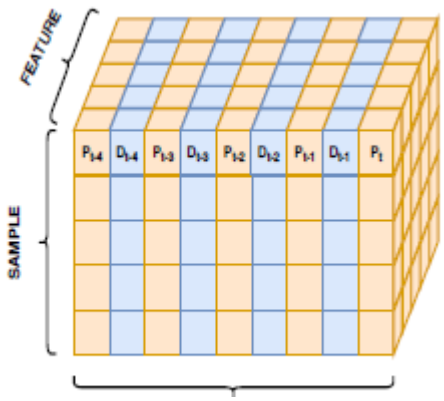


Figura 3.7: Matriu tridimensional d'entrada a la xarxa

Per escollir la mida del *TIME STEP*, s'ha realitzat un estudi amb el conjunt de carreres de la base de dades per veure quantes carreres realitzava un mateix taxista durant un interval màxim de tres hores, concloent en un nombre mitjà de gairebé cinc carreres cada tres hores.

D'aquesta forma, cada mostra del *batch* estarà formada per un conjunt de cinc parelles origen-destí on cadascuna d'aquestes parelles tindrà les seves característiques espacials, temporals i els identificadors de taxi i taxista respectius.

Si en un interval de tres hores, un mateix taxista no ha arribat a realitzar cinc carreres, les característiques de les carreres restants per arribar a cinc es completaran amb el valor de *padding* tal i com s'ha explicat en l'apartat anterior.

3.5. Xarxa Neuronal Recurrent (LSTM)

A l'Estat de l'Art s'ha explicat tota la part teòrica sobre el funcionament de les xarxes neuronals recurrents i més concretament de la LSTM. En aquest apartat també s'han explicat les diferències entre una xarxa recurrent i una xarxa convolucional, així com l'ús principal de les xarxes neuronals recurrents.

En aquest apartat, en canvi, s'explica l'aplicació de la xarxa neuronal recurrent en aquest projecte així com els diferents aspectes a tenir en compte per a la implementació d'aquesta xarxa.

Primer que tot, s'ha de tenir en compte que aquesta xarxa va precedida d'una capa d'*embedding* fet que produeix que la seva entrada no siguin valors extrets de la base de dades sinó que aquesta entrada es correspon a la sortida de la capa d'*embedding*.

Tal i com s'ha explicat anteriorment al subapartat de la capa d'*Embedding*, l'entrada de la LSTM serà la concatenació dels *latent vectors* de cadascuna de les característiques que defineixen la carrera del taxi.

A l'hora d'implementar la LSTM, la biblioteca *torch* de *Pytorch* té el seu propi mòdul LSTM fet que facilita la feina ja que no s'ha de programar tot el mecanisme intern de la cèl·lula sinó que directament només se li han d'indicar les dimensions de la xarxa, és a dir, la *input size* i la *hidden size*.

Per tant, les dimensions de sortida queden de la següent forma:

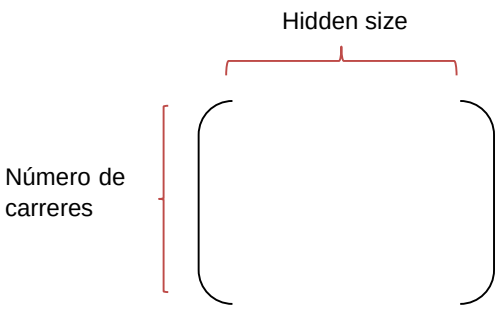


Figura 3.8: Output de la xarxa neuronal

Evidentment, aquesta matriu és tridimensional ja que té una profunditat de la mida del *batch*.

La *hidden size* es correspon al nombre de neurones que té la xarxa mentre que el número de carreres de taxi es correspon al nombre de carreres que formen cada seqüència.

Aquesta LSTM va seguida d'una capa de *Dropout* que desactiva aleatòriament una part del conjunt de neurones per tal d'eliminar possibles dependències entre neurones pròximes, aquesta és una de les mesures més utilitzades per tal de prevenir el fenomen de l'*overfitting*.

3.6. Paràmetres d'entrenament

3.6.1. Hiperparàmetres

En aquest projecte s'han dut a terme diferents entrenaments on cadascun d'ells era diferent de la resta degut al canvi de valor de diferents paràmetres del model per tal d'aconseguir un millor resultat. Els paràmetres més importants són:

- Mida del batch: és el nombre de seqüències de taxi que entra a la xarxa en una iteració de la època.
- Èpoques: és el número de vegades que una mateixa seqüència de taxi serà vista per la xarxa.
- Learning rate: és la mesura en la qual els pesos de la xarxa s'actualitzen.
- Optimitzador: element que té la funció de trobar el valor dels pesos més òptims per tenir una *loss* el més petita possible.

3.6.2. Optimitzador

En aquest projecte s'ha utilitzat l'optimitzador Adam ja que té una peculiaritat i és que ofereix una alternativa al procediment clàssic de descens de gradient estocàstic (SGD) per actualitzar els pesos de la xarxa durant l'entrenament en funció de les seves dades. Aquest optimitzador està format per la combinació dels avantatges de AdaGrad i RMSProp.

Els principals avantatges que s'enumeren a l'article són els següents:

1. Computacionalment eficient.
2. Pocs requisits de memòria.
3. Invariant a la revolució diagonal dels gradients.
4. Molt adequat en quant a problemes amb gran quantitat de dades i/o paràmetres.
5. Apropiat per problemes amb gradients molt sorollosos.

3.7. Predicció

Aquest és l'últim mòdul del model i és el que calcula el valor de les coordenades predites pel model.

La primera part d'aquest mòdul consta d'una capa Lineal i d'una capa de *Softmax*. Aquestes dues capes ens permeten passar de la matriu tridimensional de la pàgina anterior a un simple vector de longitud igual al número de clústers, és a dir, un vector de longitud 300.

Cadascun dels elements d'aquest vector es correspon a la probabilitat de cadascun dels clústers. Per tant, per obtenir la predicció de les coordenades de destí del taxi l'únic que s'ha de fer és dur a terme una combinació lineal de la sortida del *Softmax* amb el valor dels centroides de cada clúster, tal i com es pot observar a les fórmules que es mostren a continuació:

$$\tilde{y} = \sum_{i=1}^{C} P_i c_i \qquad (3.1)$$

$$P_i = \frac{exp(e_i)}{\sum_{j=i}^{C} exp(e_j)} \qquad (3.2)$$

On P_i és la probabilitat del *Softmax* i c_i és el valor del centroide del clúster *i*.

3.8. Mesures d'avaluació

Per tal d'avaluar el model i veure que no es produiex cap situació no desitjada (*overfitting/underfitting)* es defineix una funció de *loss*, que en aquest cas és el MSE (*Mean Square Error*).

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 \qquad (3.3)$$

Aquesta funció calcula l'error quadràtic mig que hi ha entre les *targets* de la base de dades (Y_i) i les coordenades predites pel propi model ($\bar{Y}_i$) per dur a terme l'algorisme de *backpropagation* i, d'aquesta forma, actualitzar els pesos del model.

Tot i això, com les coordenades es troben en graus decimals i aquestes són unes unitats no molt comuns, per fer més representatiu l'error que hi ha en la predicció del sistema complet, es realitza un canvi d'unitats mitjançant la funció de *Haversine*.

Aquesta funció permet passar de radians a kilòmetres per tal de saber la distància real que hi ha entre les coordenades correctes i les coordenades predites pel model.

La funció de *Haversine* es defineix de la següent forma:

$$d_{haversine}(p_1, p_2) = 2R\left(\sqrt{\frac{a(p_1, p_2)}{a(p_1, p_2) - 1}}\right) \quad (3.4)$$

$$a(p_1, p_2) = sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + cos(\phi_1)cos(\phi_2)sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right) \quad (3.5)$$

Sent p_1 i p_2 les coordenades correctes i les coordenades predites i on Φ_1 i λ_1 es refereixen a la latitud i la longitud de les coordenades correctes en radians i Φ_2 i λ_2 es refereixen a la latitud i la longitud de les coordenades predites en radians.

4. Resultats

En aquest apartat es mostres alguns dels resultats de les diferents proves que s'han realitzat durant els diferents entrenaments del model així com també s'especifiquen els hiperparàmetres utilitzats per obtenir cadascun dels següents resultats.

Per dur a terme els diferents entrenaments, s'ha optat per una distribució del 75% de les dades per a entrenament, 15% per a la validació i 10% per al test.

Tal i com s'ha explicat anteriorment, cada entrada de la xarxa està formada per una seqüència de 5 carreres de taxi, és a dir, 5 parelles origen-destí.

Els resultats s'han analitzat des del punt de vista de la *loss* del sistema, és a dir, l'error quadràtic mig que hi ha entre les coordenades correctes i les predites en graus decimals i també s'han calculat els kilòmetres de distància que hi ha entre aquest parell de punts per tal de facilitar la interpretació dels resultats.

4.1. Gràfiques de la loss de Train-Eval

Abans de realitzar els diferents entrenaments el primer que s'ha fet és comprovar que els conjunts de dades estan balancejats, és a dir, que cada clúster del conjunt de clústers té la mateixa proporció de punts per als conjunts utilitzats per entrenar i testejar el model.

Durant tots els entrenaments s'ha utilitzat una mida de *batch* de 64 seqüències de carreres, una capa de Dropout amb probabilitat de 0.5 i un total de 20 èpoques.

La principal característica que s'ha pogut analitzar del conjunt d'entrenaments realitzats és com la *loss* de la predicció decreix segons els valors de la *learning rate* i del número de neurones de la LSTM.

Abans d'analitzar els diferents resultats cal mencionar que al ser aquest un projecte on es prediuen unes coordenades i, tal i com s'ha dit abans, la *loss* té com a unitat els graus decimals, les gràfiques que es mostren a continuació difereixen bastant de les habituals en un problema de classificació multi-classe.

Per al primer entrenament s'han utilitzat una *learning rate* de 0.001 i 128 neurones a la LSTM. Els resultats que s'han obtingut són els següents:

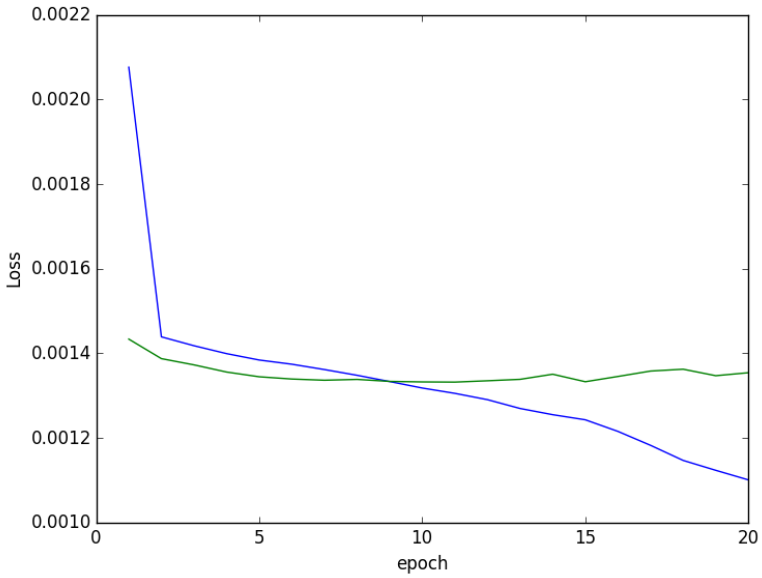


Figura 4.1: Gràfica train-eval loss model 1

S'observa com la *loss* en el conjunt de dades d'entrenament decreix de forma exponencial, tenint sobre tot un gran pendent en les primeres èpoques, fet que mostra com el model ha après. Els diferents pesos del model s'han anat actualitzat de forma que el sistema arribi a saber quina és la importància de cada clúster per predir la sortida a partir de les dades d'entrada a la xarxa.

A partir de la setena època, però, s'observa com la *loss* del conjunt de validació comença a tenir un lleuger pendent positiu que es va estenent durant les següent èpoques fet que ens porta a la situació d'overfitting.

Aquesta gràfica, des del punt de vista de l'error de predicció en kilòmetres, mostra la següent reducció de l'error, fins a la època 7 que és a partir d'on es produeix l'overfitting:

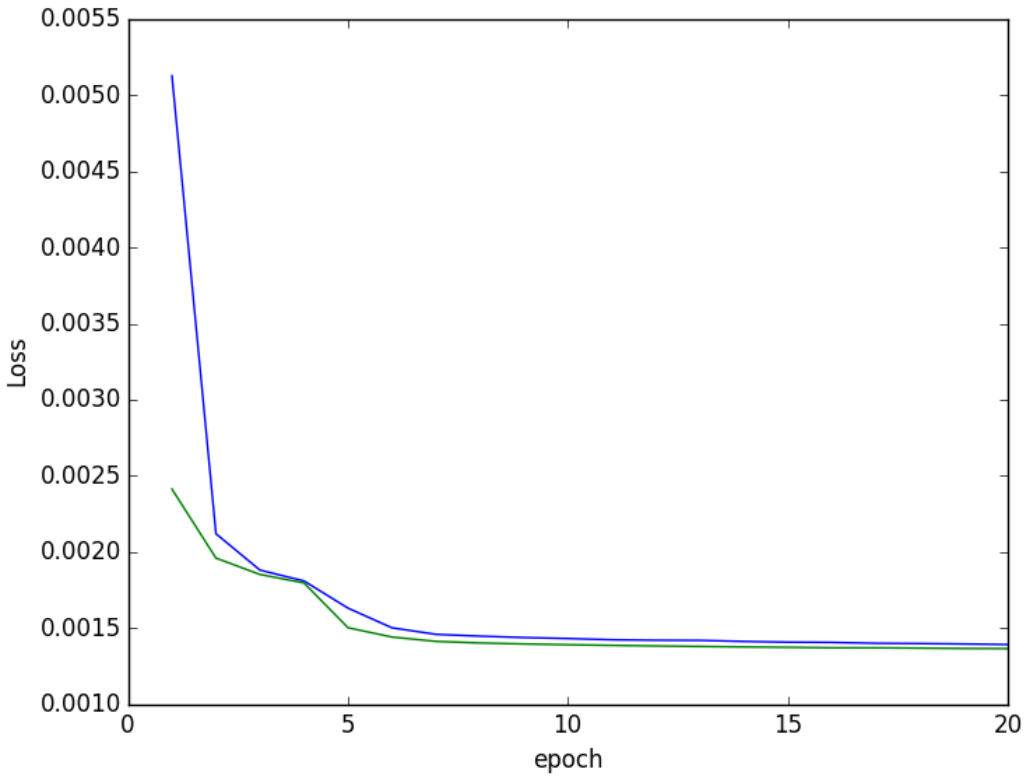
Època/Conjunt	**Train**	**Eval**
1	2.42 km	1.94 km
7	1.89 km	1.84 km
20	1.83 km	1.85 km

Taula 4.1: Error en kms model 1

Diferència (Èpoques 1-7)	0.53 km	0.1 km

Taula 4.2: Diferència en kms model 1

Per al segon entrenament, s'ha reduït la *learning rate* a 0.0001 i s'ha mantingut el mateix nombre de neurones, obtenint la següent gràfica de *loss*:



Taula 4.3: Gràfica train-eval loss model 2

S'observa com el canvi en la *learning rate* provoca que l'entrenament sigui més lent però es pot veure com el valor final de la *loss* del conjunt de validació acaba sent pràcticament el mateix.

També s'observa que en cap moment s'arriba a una situació d'overfit però si que a partir de la 6-7 època el valor de la *loss* de validació disminueix molt lentament quedant així una línia pràcticament recta. Tot i això si s'augmentés el nombre d'èpoques, s'arribaria a un valor de *loss* més baix que el de l'època 20.

Època/Conjunt	Train	Eval
1	5.33 km	2.92 km
7	1.95 km	1.92 km
20	1.91 km	1.89 km

Taula 4.4: Error en kms model 2

Diferència (Èpoques 1-7)	3.38 km	1 km

Taula 4.5: Diferència en kms model 2

En cas de disminuir la probabilitat de la capa de Dropout de 0.5 a 0.3 les neurones passen a tenir més dependència de les neurones pròximes i això permet a la xarxa aprendre d'una forma més ràpida però també pot crear una situació d'overfit sobre el conjunt d'entrenament. En el cas d'aquest projecte, la diferència entre utilitzar una probabilitat de 0.5 o una de 0.3 a la capa de Dropout és mínima ja que l'única diferencia notable és que s'arriba a un valor estable de *loss* a la sisena època en lloc de a la setena, com ocorria amb la probabilitat de 0.5 .

A l'augmentar el número de neurones de la LSTM fins a un valor de 512 neurones, s'observa com al principi la xarxa aprèn més ràpidament ja que els valors de la *loss* són inferiors als de la *loss* quan la LSTM consta de 128 neurones però a partir de la cinquena època comença a haver overfit ja que els valors de *loss* en el conjunt de validació tenen una lleugera tendència a augmentar mentre que en el conjunt d'entrenament encara continuen baixant.

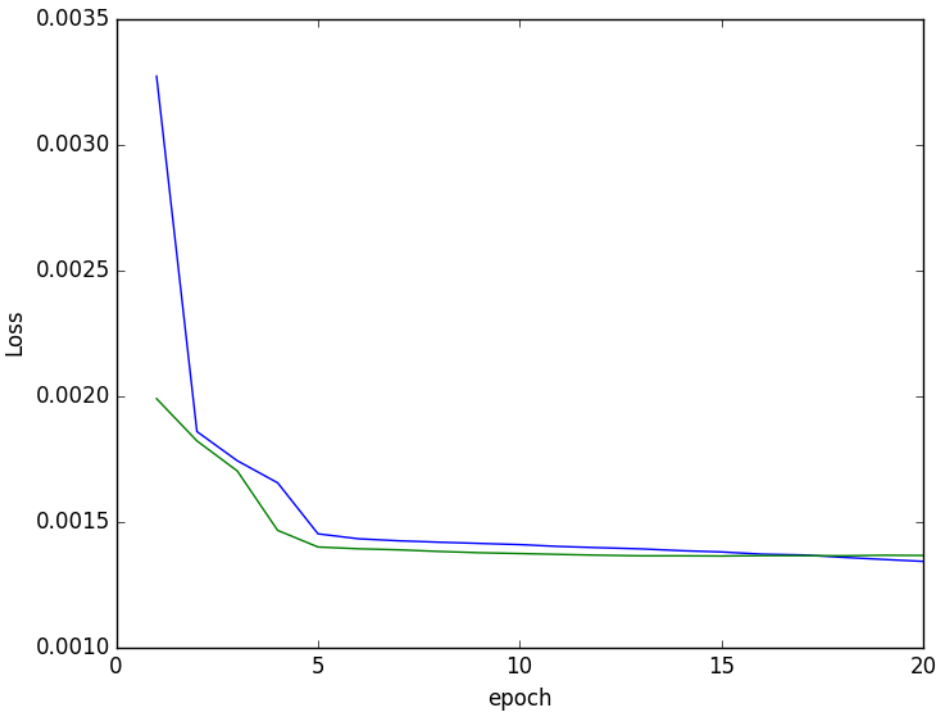


Figura 4.2: Gràfica train-eval loss model 3

Època/Conjunt	Train	Eval
1	3.68 km	2.37 km
7	1.93 km	1.91 km
20	1.90 km	1.89 km

Taula 4.6: Error en kms model 3

Diferència (Èpoques 1-7)	1.75 km	0.46 km

Taula 4.7: Diferència en kms model 3

Finalment, s'ha realitzat el test del sistema general. Per a aquest test s'ha escollit el model de la setena època que té els següents hiperparàmetres:

- *Learning rate*: 0.001
- Número de neurones: 128
- Probabilitat capa de Dropout: 0.5
- Número d'èpoques: 20
- Mida del *batch*: 64

Amb aquest model, s'obtenen uns resultats sobre el conjunt de test tals que la distància d'error mitjana en kilòmetres entre les *targets* i les coordenades predites és de 1.72 kilòmetres.

Aquest resultat es pot comparar amb resultats publicats en un altre article que té el mateix objectiu que aquest projecte però en altres ciutats [8].

Model	Porto	San Francisco	Manhattan
NN	3.215	3.023	2.375
MMLP	3.211	1.994	$2.543^{(*)}$
MMLP-SEQ	3.003	2.762	2.554
LSTM	2.923	2.547	2.111
LSTM (BOC)	2.923	2.397	**2.085**
LSTM (BOC+W2V)	**2.88**	**2.270**	2.088

Figura 4.3: Taula de resultats

El resultat d'aquest projecte és comparable amb els resultats de la quarta fila d'aquesta taula. Així doncs, s'observa com el model òptim d'aquest projecte obté uns resultats millors que els models aplicats a les ciutats de Porto, San Francisco i Manhattan.

5. Pressupost

El projecte s'ha realitzat utilitzant els recursos facilitats pel Grup d'Imatge de la UPC, per tant els costos de manteniment i de materials és zero. Però si aquest projecte s'hagués realitzat sense els recursos del Grup d'Imatge de la UPC, el cost setmanal d'un servidor suficientment potent per poder realitzar aquest projecte costaria uns 470€ setmanals.

La base de dades proporcionada pel grup CARNET també ha estat proporcionada de forma gratuïta.

Això significa que els únics costos que s'han produït durant el desenvolupament d'aquest projecte són els salaris dels investigadors que han treballat en ell. Tenint en compte aquests aspectes, la taula de costos queda configurada de la següent forma:

	Persones	**Salari/hora**	**Hores/Setmana**	**Setmanes**	**Total**
Enginyer júnior	1	8,00€	25h	21	4200€
Enginyer sènior	2	60,00€	2h	21	5040€
Servidor	470€/setmana			21	9870€
				Total	19110€

Taula 5.1: Costos del projecte

6. Conclusions i futurs desenvolupaments

L'objectiu d'aquest projecte ha estat estudiar i investigar sobre la mobilitat dels taxis i dels *car sharing* a la ciutat de Barcelona per tal d'intentar reduir els kilòmetres innecessaris que realitzen aquests durant el dia a dia degut a que aquest fet permetria una reducció de la contaminació tant acústica com medi ambiental i també permetria als conductors evitar possibles rutes amb més trànsit i fer que la seva conducció sigui menys pesada.

S'ha observat que es poden obtenir uns resultats amb una precisió considerable tenint en compte el conjunt d'informació de la base de dades ja que no es disposava de cap tipus de dades descriptives de cadascun dels clústers utilitzats com a entrada del sistema (*Bag of Concepts*) com per exemple si que utilitzaven en un dels articles llegits prèviament a l'inici del desenvolupament del projecte.

També s'ha de tenir en compte que en aquest projecte només s'utilitza la informació del punt de sortida i no de diferents punts de la trajectòria, fet que ajudaria a obtenir millors resultats ja que la xarxa tindria més informació durant l'entrenament però per altra banda no es podria saber la predicció des de l'inici de la trajectòria.

L'ampliació de la base de dades tant en nombre de carreres de taxi com en contingut d'informació que descrigui les característiques espacials del lloc per on transita el vehicle seria una gran ajuda per a una posterior millora del funcionament del sistema un cop ja s'ha realitzat un primer estudi sobre el tema i s'han obtingut uns resultats adequats però, al mateix temps, millorables ja que com s'ha vist a l'apartat on s'exposen els diferents resultats obtinguts, el sistema aprèn ràpidament fins a una precisió d'aproximadament 1.8 kms però posteriorment la millora és imperceptible, de l'ordre d'algun metre, i hi ha un lleuger overfit sobre les dades d'entrenament.

Un futur desenvolupament a realitzar podria ser la implementació d'una aplicació per a mòbil per a que els diferents conductors tinguin en compte quina seria la ruta més adequada a seguir o per saber cap a on dirigir-se per buscar clients sense tenir que recórrer més kilòmetres dels necessaris ja que al només tenir en compte la informació del punt d'origen, el càlcul seria instantani i s'obtindria la informació necessària al moment.

Bibliografia

[1] M. Gonzalez, C. Hidalgo y A. Barabasi, «Understanding individual human mobility patterns,» *nature,* nº 453, p. 779, 2008.

[2] M. Ahmed, G. Barlacchi, S. Braghin, F. Calabrese, M. Ferretti, V. Lonij, R. Nair, R. Novack, J. Paraszczak y A. Toor, «A multi-scale approach to data-driven mass migration analysis,» *SoGood ECML-PKDD,* 2016.

[3] G. Barlacchi, C. Perentis, A. Mehrotra, M. Musolesi y B. Lepri, «Are you getting sick? predicting influenza-like symptoms using human mobility behaviors,» *EPJ Data Science,* nº 6, p. 27, 2017.

[4] L. Pappalardo, F. Simini, S. Rinzivillo, D. Pedreschi, F. Giannotti y A. Barabási, «Returners and explorers dichotomy in human mobility,» *Natures communications,* nº 6, p. 8166, 2015.

[5] A. De Brébisson, É. Simon, A. Auvolat, P. Vincent y Y. Bengio, «Artificial neural networks applied to taxi destination prediction,» *arXiv preprint arXiv:1508.00021,* 2015.

[6] D. Yao, C. Zhang, J. Huang y J. Bi, «Serm: A recurrent model for next location prediction in semantic trajectories,» *ACM,* pp. 2411-2414, 2017.

[7] F. Gers, J. Schmidhuber y F. Cummins, «Learning to forget: Continual prediction with lstm,» 1999.

[8] A. Rossi, G. Barlacchi, M. Bianchini y B. Lepri, «Modeling taxi drivers' behaviour for the next destination prediction,» *arXiv:1807.08173,* 2019.

[9] «https://colah.github.io/posts/2015-08-Understanding-LSTMs/,» 27 08 2015. [En línea].

[10] T. Mikolov, K. Chen, G. Corrado y J. Dean, «Efficient estimation of word representations in vector space,» *arXiv:1301.3781v3,* 2013.

Glossari

DL: Deep Learning

RNN: Recurrent Neural Network

LSTM: Long Short-Term Memory